

CPE — Tópico 05

Comandos de Repetição: `while`, `do-while` e `for`

Departamento de Engenharia Elétrica – UnB

- 1 Comandos while e do-while
 - while (condicao) { comandos }
 - do { comandos } while (condicao);
 - Exemplos

- 2 Comando for
 - for (inicio ; condicao ; passo) { comandos ; }
 - Exemplos

Roteiro

- 1 Comandos while e do-while
 - while (condicao) { comandos }
 - do { comandos } while (condicao);
 - Exemplos
- 2 Comando for
 - for (inicio ; condicao ; passo) { comandos ; }
 - Exemplos

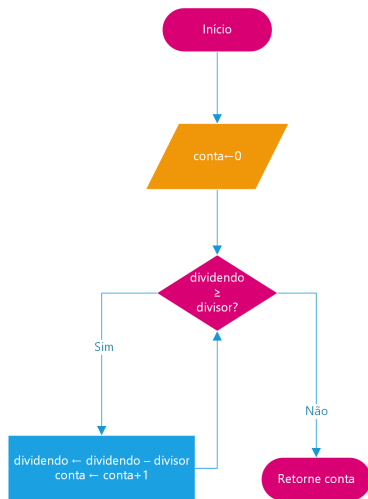
Introdução

- Até agora, vimos como escrever programas capazes de executar comandos de forma linear, e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, eventualmente é necessário executar um bloco de comandos várias vezes para obter o resultado.

Exemplo

Calcule a divisão inteira de dois números usando apenas soma e subtração

Algoritmo



Repare que conta resulta na divisão inteira de dividendo por divisor

Algoritmo

- Três variáveis: dividendo, conta, divisor
 - 1 $\text{conta} \leftarrow 0;$
 - 2 **Repita** enquanto $\text{dividendo} \geq \text{divisor}$
 - 1 $\text{dividendo} \leftarrow \text{dividendo} - \text{divisor}$
 - 2 $\text{conta} \leftarrow \text{conta} + 1$
 - 3 Exiba conta

Introdução

- Para cada comparação, fazemos:
 - ① subtraímos divisor de dividendo.
 - ② incrementamos conta
- Quando divisor supera dividendo, as demais comparações retornariam falso e não se executaria o bloco de comandos.

Introdução

- Seria, portanto, interessante existir um comando da linguagem que repetisse a comparação e executasse os comandos dentro enquanto a condição dada fosse verdadeira

```
/* Enquanto é verdade que dividendo>=divisor, execute */  
{  
    dividendo = dividendo - divisor;  
    conta++;  
}
```



```
while (condicao) { comandos }
```

- Estrutura:

```
while ( condicao )
```

```
    comando;
```

ou

```
while ( condicao ) {
```

```
    comando1;
```

```
    comando2;
```

```
    ...
```

```
    comandoN;
```

```
}
```

- Enquanto a condição for **verdadeira**, ele executa o(s) comando(s);

Divisão inteira

```
1 #include <iostream>
2 using namespace std;
3 //programa que calcula a divisao inteira
4 int main() {
5     int dividendo, divisor;
6     int conta;
7
8     cout << "Dividendo: ";
9     cin >> dividendo; //entrada
10    cout << "Divisor: ";
11    cin >> divisor; //entrada
12
13    conta = 0;
14    while (dividendo >= divisor) { //repeticao
15        conta++;
16        dividendo = dividendo - divisor;
17    }
18
19    cout << "Resultado: " << conta
20         << "\nResto: " << dividendo << endl;
21    return 0;
22 }
```

Imprimindo os 100 primeiros números inteiros, separados por espaço

```
int i=1;
while (i<=100)
{
    cout << i << ' ';
    i++;
}
```

Imprimindo os n primeiros números inteiros

```
int i=1,n;  
cin >> n;  
while (i<=n)  
{  
    cout << i << ' '  
    i++;  
}
```

Imprimindo as n primeiras potências de 2

```
int i=1, n, pot=2;
cin >> n;
while (i<=n)
{
    cout << "2^" << i << " = " << pot << endl;
    i++;
    pot *= 2;
}
```

```
while (condicao) { comandos }
```

- 1. O que acontece se a condição for falsa na primeira vez?
a=-1;
while (a>0) a=a-1;
- 2. O que acontece se a condição for sempre verdadeira?
while (a==a) a=a+1;

```
while (condicao) { comandos }
```

- 1. O que acontece se a condição for falsa na primeira vez?

```
a=-1;
```

```
while (a>0) a=a-1;
```

R: Não executa $a=a-1$ i.e. programa nunca entra na repetição (*loop*).

- 2. O que acontece se a condição for sempre verdadeira?

```
while (a==a) a=a+1;
```

R: Executa $a=a+1$ indefinidamente i.e. programa entra na repetição e nunca sai (*loop* infinito).

```
while (condicao) { comandos }
```

- Estudando a estrutura “normal” do while mais a fundo.

```
while (i<=n) ← condição de repetição  
{  
    cout << i;  
    i++; ← Comando de passo  
}
```

- O oposto (negação) da condição de repetição é conhecida como **condição de parada**:
 $!(i \leq n) \Rightarrow i > n$ é a condição de parada, neste exemplo.


```
while (condicao) { comandos }
```

- *loop de fim determinado*

```
cin >> preco;
while (i<=n) {
    total = total + preco;
    i++;
    cin >> preco;
}
```

- *loop de fim indeterminado*

```
cin >> preco;
while (preco>0) {
    total = total + preco;
    cin >> preco;
}
```

```
do { comandos } while (condicao);
```

- Estrutura:

```
do
    comando;
while ( condicao );
ou
do {
    comando1;
    comando2;
    ...
    comandoN;
} while ( condicao );
```

- Diferença do while: **sempre** entra na primeira vez.

Máximo Divisor Comum - MDC(x,y)

Supondo, sem perda de generalidade, que $x \geq y$, o $\text{MDC}(x, y)$ é definido da seguinte forma:

$$\text{MDC}(x, y) = \begin{cases} y & \text{caso } x \bmod y = 0 \\ \text{MDC}(y, x \bmod y) & \text{caso contrário} \end{cases}$$

MDC(x,y)

```
/* assume-se aqui que x > y */  
do  
{  
    r = x % y;  
    x = y;  
    y = r;  
} while (r!=0);
```

- Repare que r só é calculado dentro do *loop*
- Veja exemplo em `mdc-completo.cpp`

Soma de n valores inteiros

```
soma = 0;
while (n > 0) {
    cout << "numero a ser somado: ";
    cin >> parcela;
    soma += parcela;
    n--;
}
cout << "Soma: " << soma << endl;
```

- Veja exemplo em soma-n.cpp

Soma até 0

```
soma = 0;
cout << "numero a ser somado (0 para sair): ";
cin >> parcela;

while (parcela != 0) {
    soma += parcela;
    cout << "numero a ser somado (0 para sair): ";
    cin >> parcela;
}

cout << "Soma: " << soma << endl;
```

- Veja exemplo em soma-ate-0.cpp

Soma até 0

```
soma = 0;

do {
    cout << "numero a ser somado (0 para sair): ";
    cin >> parcela;
    soma += parcela;
} while (parcela != 0);

cout << "Soma: " << soma << endl;
```

- Veja exemplo em `soma-ate-0-do-while.cpp`

Roteiro

- 1 Comandos while e do-while
 - while (condicao) { comandos }
 - do { comandos } while (condicao);
 - Exemplos

- 2 Comando for
 - for (inicio ; condicao ; passo) { comandos ; }
 - Exemplos

Uso comum de comandos de repetição

```
i = 0;
while (i < n) {
    /* Vários comandos */
    i++;
}
```

Problemas do `while` e do `do ... while`

- Onde são inicializadas as variáveis usadas na condição do *loop*?
- O passo pode estar em qualquer ponto do loop.

Apenas a **condição** está destacada no contexto.

• Ex.:

```
i = 0;
```

← Inicialização de i

```
/* várias linhas de código */
```

```
while (i < 10) {
```

← Condição de loop

```
    j = j * 2;
```

```
    l = j - i;
```

```
    i++;
```

← Passo

```
    k = i + j;
```

```
}
```

```
for (início ; condicao ; passo) { comandos }
```

- Estrutura:

```
for (início ; condicao ; passo)  
    comando;
```

ou

```
for (início ; condicao ; passo) {  
    comando1;  
    comando2;  
    ...  
    comandoN;  
}
```

- Início: Uma ou mais atribuições, separadas por “,”
- Condição: Idêntico ao while
- Passo: Um ou mais comandos, separados por “,”

Exemplo

- Programa com while

```
i = 0;
while(i < n){
    cout << i << endl;
    i++;
}
```

Exemplo

- Programa com for

```
for (i = 0 ; i < n ; i++) {  
    cout << i << endl;  
}
```

```
for (inicio ; condicao ; passo) { comandos ; }
```

Quando usar for? Quando usar while?

- Em termos de implementação, ambos são intercambiáveis, porém...
- Em termos de escrita de código claro, há uma diferença:
 - *Loop* de fim determinado
“Para c=1 até c=100, faça”
 - *Loop* de fim indeterminado
“Enquanto não digitar enter, continue lendo”

```
for (inicio ; condicao ; passo) { comandos ; }
```

Quando usar for? Quando usar while?

- Em termos de implementação, ambos são intercambiáveis, porém...
- Em termos de escrita de código claro, há uma diferença:
 - *Loop* de fim determinado
“Para c=1 até c=100, faça” ← recomenda-se usar **for**
 - *Loop* de fim indeterminado
“Enquanto não digitar enter, continue lendo” ← recomenda-se usar **while** ou **do-while**

I'll not throw paper airplanes in class

```
#include <stdio.h>
int main(void)
{
    int count;

    for(count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");

    return 0;
}
```

AVISO 10-3



Como imprimir os n primeiros números ímpares?

```
impar = 1;
for(i = 0; i < n; i++) {
    cout << impar << ' ';
    impar += 2;
}
```

- Veja exemplo em `n-impares-for.cpp`

Como imprimir os n primeiros números ímpares?

```
for (i = 0, impar = 1; i < n; i++, impar += 2)
    cout << impar << ' ';
```

Inicialização e/ou atualização podem ter vários comandos separados por vírgulas!

- Veja exemplo em `n-impares2-for.cpp`

Fatorial

```
cout << "Entre com um numero inteiro positivo: ";
cin >> n;

fat = 1;
for (i = 2; i <= n; i++)
    fat *= i;

cout << "O fatorial de " << n
    << " e igual a " << fat
    << endl;
```

O que acontece com números muito grandes?

- Veja exemplo em `fatorial.cpp`

Como imprimir uma linha de '*'s usando o comando for

```
*****
```

```
for (i = 0; i < n; i++)  
    cout << "*";  
cout << "\n";
```

- Veja exemplo em `linha-for.cpp`

Como calcular o maior número entre n lidos?

```
cin >> n;
maior = INT_MIN;
for(i=1; i <= n; i++) {
    cout << "Entre com um inteiro: ";
    cin >> num;
    if (num > maior)
        maior = num;
}
cout << "O maior inteiro lido foi " << maior << endl;
```

- Veja exemplo em maiorn.cpp

Dilbert

